SWARM

Lineage

- Written for SimCity 4

  - Drove all disasters, demolition, UI, ambient, vehicles
- Sims 2

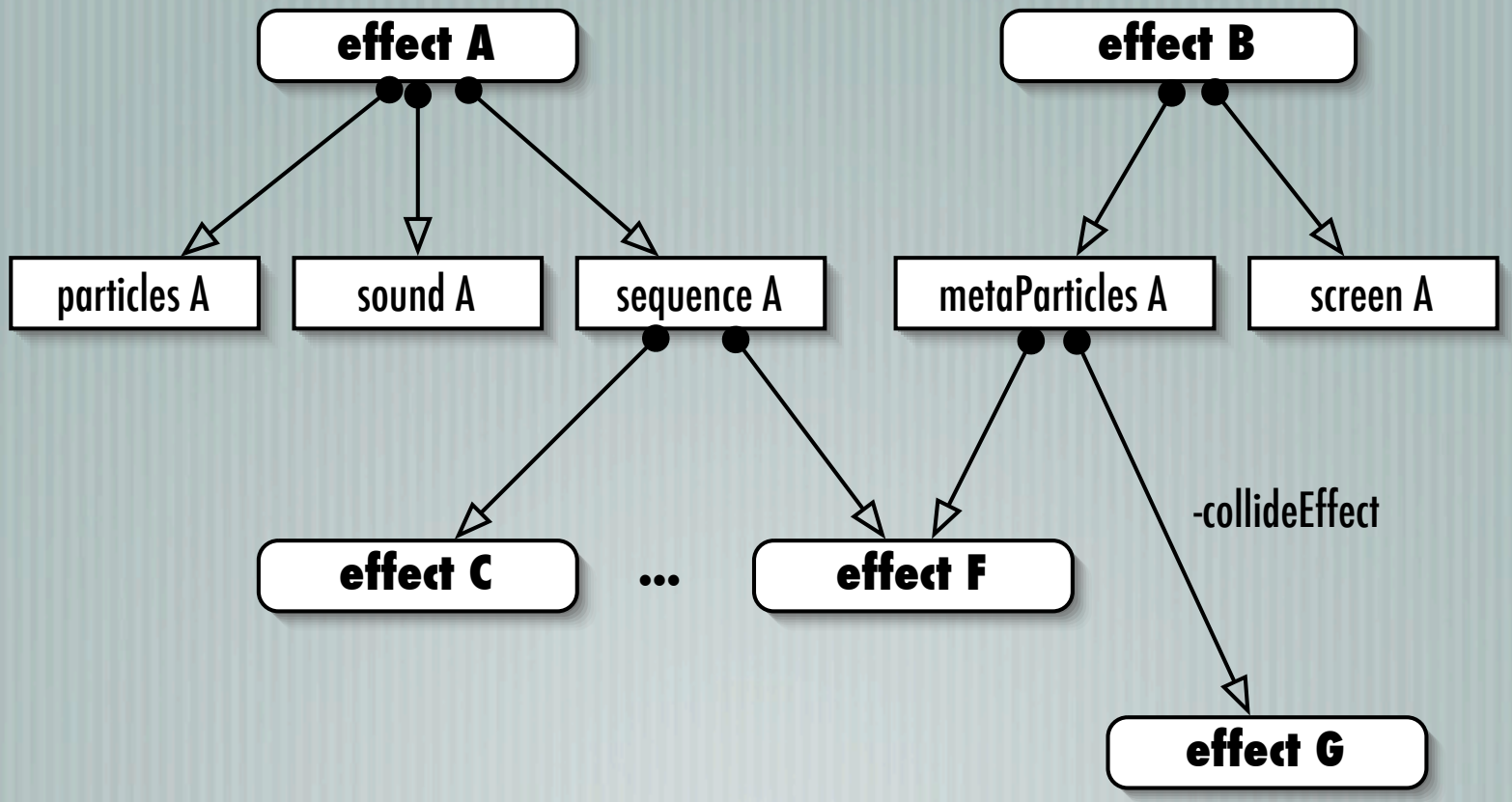
  - Object effects, UI, neighbourhood, camera moves
- Spore

  - 70% of GDC demo

Main Features

- Data scripted with hot loading
  - Not interpreted: no run-time script parsing in final game
- Component based, extensible
  - Particles, virtual particles, models, sound, decals ...
- Hierarchical
- Code control: Set paths, control parameters

Architecture



Scripting

- Very simple command-based system
- Inspired by scripts used in animation production (R&H, CMU)
- Easy to add options and new commands

```
command arg1 arg2 -flag -option optionArg1
```

```
block
    ...
end
```

Scripting

```
particles color_test
   life 2
   source -square 16

   rate 200
   emit -speed 200 -dir (0, 1, 0)

   color (0, 0, 1) (1, 0, 0) (0, 1, 0) (0, 0, 0) (1, 1, 1)
   alpha 1
   size 0.2

   texture snowflake
end
```

- Shows colour curve: almost any variable can be specified as a curve

Curves Demo

Script Conveniences

- Inheritance
  - Variants of base system
- Macros
  - Instantiate suite
- Variables
  - Switch on platform, overall properties

```
particles blue_flames : base_flames
   color (0, 0, 1)
end
```

Demo

Grouping

- An effect can include a number of components, or other effects (nesting)
- A single component (e.g. fire_particles) can be shared amongst many effects
- When a component is included, you specify modifiers:
  - Position, orientation, scale
  - Applicability

Selection

- List a block of components, optionally assigning each a property
- When an effect of that type is created, one of the components will be selected
- Useful for establishing sets/grammars for procedural effects

```
select
   model honda_civic -prob 0.5
   model toyota_prius -prob 0.1
   model ford_behemoth_SUV -prob 0.4
end
```

Particles

- Standard textured quads
- Curves: colour, alpha, aspect, size, etc.
- Forces, collision
- Various emitter shapes
- Stretch: cheap motion blur
- Random walk force

Meta Particles

- Each “meta” particle is itself an effect
- Key to a lot of interesting behaviours
- Per-particle random walks, multiple walks
- Each particle has full coordinate frame
- Paths: Set one or more of speed/velocity, time at each point

Maps

- Sourced from image or the game
  - Control particle density
  - Select range: 0.2 – 0.5
  - Control emission colour
  - Treat as height field for collision/repulsion
  - Combine via map operators

Sequencing

- Straight forward: play a series of effects with timing
  - Play for x seconds
  - Or just wait for it to finish
- Also: particle sequences
  - When particle in first system dies, it's moved to the second
  - Cheap way to get evolving behaviour over time

Camera Control

- Grab camera and slave to an effect
  - E.g. to a meta particle following a flythrough path
- Control orientation, pitch
- Switch between camera controller modes
- Use in conjunction with...

Screen Effects

- Simple overlays/backgrounds
- Cross fades
- Screen shake
- Filters: blur, bloom, distort, motion trail, etc.
- Artist controls strength, tinting, can even compose filters a la photoshop

Screen Demo

More ...

- Models: Use curves to drive animation channels
- Lights
- Sounds
- Game interaction
  - Send game messages
- Generic collision surfaces, built-in or from game

More ...

- Game-specific components
  - SimCity: Scrubber, for demolition
  - Sims 2: Deformable water mesh
  - Spore: Creature painting brushes
- Easy to drop in own external component, add new commands
- Leverage existing components: adding new tool attachments

Adaption

- Can switch components on and off:
  - Level of detail changes
  - Application effect flag changes
  - Set state of an effect from code
- Modify size, scale, alpha according to fractional LOD

Level of Detail

- Global LOD: set by application
  - Usually related to elevation: distance from ground
- Camera LOD: relative to camera placement
  - Artist controls LOD distances
  - Also alpha, size

Parameters

```
enum tFloatParamType
{
   kParamEmitStrength,        ///< 1 float:       scales the emit velocity
   kParamRateScale,           ///< 1 float:       scales emit rate

   kParamTargetPoint,         ///< 1 vector:      world space position

   kParamAttractorLocation,   ///< 1 vector:      position
   kParamAttractorIntensity,  ///< 1 float:       intensity, 0-1

   kParamViewRay,             ///< 2 vectors:     position & direction
};

enum tUnknownParamType
{
   kParamMessageData,         ///< IUnknown:          data for any messages sent.
   kParamTransformNode,       ///< ITransformNode:    transform node to attach this effect to.
};

virtual bool SetVectorParams(tFloatParamType   id, const tTSVector3* params, int count = 1) = 0;
virtual bool SetFloatParams (tFloatParamType   id, const Float32*    params, int count = 1) = 0;
virtual bool SetUnknownParam(tUnknownParamType id, cIGZUnknown*      param) = 0;
///< Used for passing data into an effect from code. Returns true if the data was handled.
```

Motivation

- Why isn't this overkill?
- Compartmentalizes complexity: Artists can make elaborate effects without lots of custom code. No waiting!
- Faster turnaround & iteration
- It's an amplifier on artist creativity
- Enables artists to make procedural content

Proceduralism

- Proceduralism lets artists proceed beyond merely providing individual pieces of content
- Create artistic systems & processes
- Any artist can model a couch: Baseline modeling & animation is being commoditized.
- To stay current, artists need to think about how they operate in context of larger systems, and participate in their design

Key Advantages

- Gives artists direct entry into the game
  - Not just throwing art over the wall to engineers
  - Immediately viewable
- Has worked well even for non-technical artists
- Fast feedback loop between artist / FX engineer

Data Scripting

- For Artist-driven scripting of procedural systems, avoid logic scripting!
  - Too much unnecessary, irrelevant complexity
  - Too easy to cause performance problems
  - Maintenability nightmare
- Can do a lot with data scripting